

QoS-Aware Core Migration for Efficient Multicast in Mobile Ad hoc Networks^{*}

Manish Kochhal^{a,*} Loren Schwiebert^b Sandeep Gupta^c Changli Jiao^a

^a*Department of Electrical and Computer Engineering*

^b*Department of Computer Science*

Wayne State University, Detroit, MI 48202, USA

^c*Department of Computer Science and Engineering*
Arizona State University, Tempe, AZ 85287, USA

Abstract

A mobile ad hoc network (MANET) comprises a group of randomly wandering wireless-enabled portable computing devices that in the absence of a centralized base station and limited transmission range communicate with distant nodes through multihop paths. Random node mobility and the unpredictable characteristics of the wireless medium make application desired QoS provisioning in MANETs very difficult. Hence instead of satisfying QoS requirements on an absolute basis, it is viable to provide QoS on a best effort basis with additional protocol support for gradual QoS adaptations to unpredictable MANET dynamics. Our proposal identifies this design philosophy and implements it over a popular tree-based multicast protocol (MAODV) for MANETs.

In a tree-based multicast, the core is always a flashpoint for almost all the traffic that flows in the network. The relative placement of the core with respect to the group members affects the delay experienced by leaves of the tree thereby influencing the performance of the multicast. Our protocol therefore seeks to conduct an optimal core selection by having the core record the history of delays to group members in terms of the relative time difference between sending the multicast packets and receiving corresponding acknowledgments from the respective subtree branches. If the average delay exceeds the QoS requirement by a given threshold, the core selects a better core candidate from nearby members to reduce migration overhead. In this way, the core migrates on a hop-by-hop basis continuously adapting to the network dynamics; and whenever the topology remains constant for a sufficient duration, the core reaches an optimal position corresponding to the desired QoS.

Key words: MANETs, QoS, tree-based multicast (MAODV), packet delay history, delay threshold, and group-leader (core) migration.

1 Introduction

Wireless enabled portables [4] [25] [43] are becoming increasingly popular due to their decreasing cost, small form factor, and their ability to communicate in the absence of wires. Also, with the advancement in VLSI integration, low power computing, and battery technologies, these devices have found new uses in traditional application areas such as civil, military, medical, and environmental domains. Typical examples include providing support for communications in battlefields, fire rescue missions, and microclimate monitoring using sensor-enabled wireless devices.

A network is formed on-demand, if such devices are in the vicinity and they wish to communicate for collaborative computing. In the absence of a centralized base station, such an ad hoc network depends completely upon the selfless cooperation of its users to distributively support standard protocol services such as communication scheduling and routing. For example, in figure 1, due to the limited transmission range distant nodes communicate via intermediate nodes that cooperate to relay packets in a multihop fashion. Examples of wireless ad hoc networks include wireless sensor networks (WSNs) [32] and mobile ad hoc networks (MANETs) [25] [30].

A mobile ad hoc network makes protocol solutions to distributed networking services difficult and complex [18]. This is because the communicating nodes are given an additional degree of freedom, which is uncontrollable and unpredictable mobility. With node movements, the network topology shown in figure 1 will change dynamically as new links are formed and existing links break. To add to this, the variability of the wireless communications medium with respect to time and environment [34], such as the fluctuations in signal strength and propagation, adds a challenge much more difficult than the traditional wired networks experience.

The ease of deployment of MANETs makes it a practical choice for a variety of applications despite their natural limitations. Applications on the Internet usually involve point-to-point communication whereas in MANETs application scenarios dictate nodes to participate in sharing information among the group in one-to-many or many-to-many communication patterns [26]. Flooding [19], broadcast [28], multicast [16], and anycast [29] are the network services that provide such group communication patterns [15]. Efficient support

* Expanded version of a short preliminary paper [21] that appeared in IPCCC 2002, Phoenix, Arizona, USA.

* Corresponding author.

Email addresses: manishk@wayne.edu (Manish Kochhal), loren@wayne.edu (Loren Schwiebert), sandeep.gupta@asu.edu (Sandeep Gupta), changlijiao@yahoo.com (Changli Jiao).

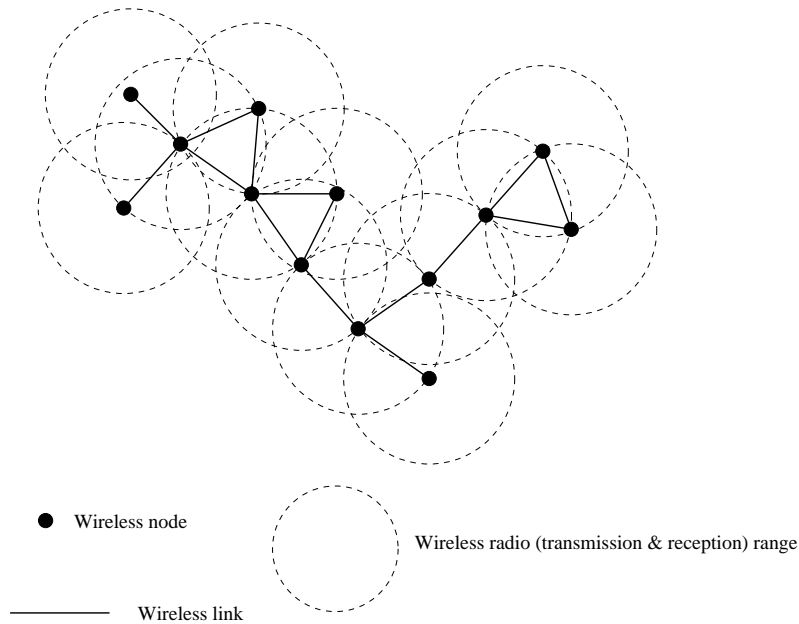


Fig. 1. Wireless ad hoc network.

for such services becomes critical depending upon the application requirements. For example, the situational-awareness of a fire-rescue operation may depend upon the delay experienced by a group of rescuers in sharing critical event information. In other words, applications for MANETs demand a certain quality of service (*QoS*) [8] qualified in the above example by a metric such as *delay*.

QoS is usually defined as a set of service requirements that the network needs to satisfy while routing data from a source to its intended recipient(s). The network is expected to guarantee a pre-specified set of quantifiable service qualities to users in terms of end-to-end performance such as delay, bandwidth, delay jitter, energy efficiency, service coverage, etc. QoS provisioning in a dynamic MANET environment is very difficult [8]. Since hard guarantees on service quality are impossible in MANETs, some researchers have proposed the notion of *soft QoS* [40]. Soft QoS means that after connection set up, there may exist transient periods of time when the QoS specifications are not met. During these intervals of QoS disruption, MANET protocols usually try to achieve best-effort QoS.

Multicasting is widely used for group communication and has received a great deal of attention in recent years. In multicast, the sender transmits a single message that is replicated within the network and delivered to multiple recipients. For this reason, a multicast message typically requires less total bandwidth than sending a separate message to each recipient. The sender and receivers form the group sharing this information. There are different approaches to providing multicast services in a MANET and the control overhead

in all of these primarily depends on the underlying network organization used for group communication [41]. Flooding, mesh, cluster, and tree represent the various approaches for multicast. Flooding is a very simple approach and it requires no underlying network organization. However, it results in a lot of redundancy in terms of message transmissions and receptions. Mesh, cluster, and tree-based approaches limit this redundancy by strictly following their fundamental network organization while transmitting and replicating multicast messages. Although these generate minimal data traffic, the amount of control traffic exchanged for updates and maintenance of the underlying network organization is very high. With high node mobility and an increasing number of nodes these approaches do not scale well in comparison to flooding which seems to be the only viable option in such situations.

In this paper, our goal is to make a multicast protocol QoS sensitive according to MANET dynamics. QoS support in MANETs can encompass optimizations and sharing of state information across many layers of the protocol stack [9] [12]. However, we are interested in using the information available at the network layer of a tree-based multicast routing protocol with the application-layer providing hints about the desired QoS *delay threshold* for multicast data delivery. To this end, we propose a QoS-aware core migration approach for multicast protocols that uses the elementary core-based tree (CBT) primitive for constructing and maintaining a group-shared multicast tree. In general, our approach is applicable to hierarchical type multicast routing protocols in which most of the traffic flows through some node(s) that assume a higher-level network-management responsibility, for example the role of a group leader of a tree or cluster.

Multicast tree-based routing protocols such as MAODV (Multicast Ad hoc Distance Vector) [35] [31] usually start by performing a route discovery cycle consisting of two elementary messages such as a route request and route reply (RREQ and RREP). Multicast group membership is dynamic; nodes are able to join and leave the group at any time. One of the multicast group members (usually the first source) randomly assumes the responsibility of a group-leader (core) and it starts broadcasting RREQs. On receiving RREQs, other group members then unicast RREPs along the reverse shortest path to the core. These and other new connections to the tree in the form of RREPs are accepted either by existing members or by the core. Non-member nodes that fall along this reverse path and connect two disjoint members are also accepted as part of the forwarding tree. Thus, as nodes join the group, a single bidirectional delivery tree composed of multicast group members and nodes used to connect them is created. If the source is not the group leader then in that case multicast data for a particular group are first forwarded as unicast messages toward the core until they reach a router that belongs to the corresponding delivery tree. The data packets are then forwarded to all outgoing interfaces that are a part of the delivery tree except the incoming

interface. This is illustrated in figure 2.

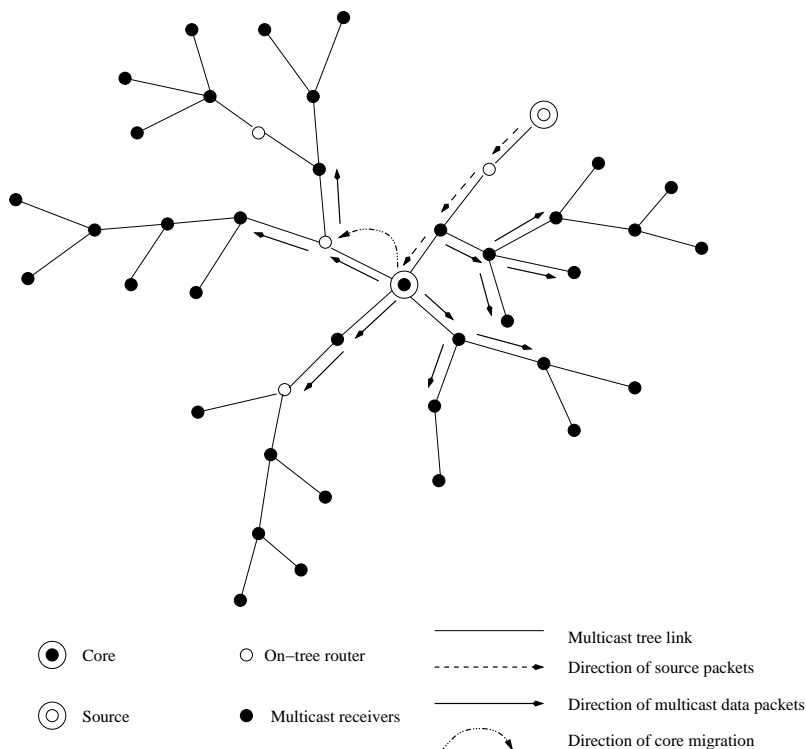


Fig. 2. Core Based Multicast Tree

The tree used for forwarding the multicast messages to a particular group is a single tree regardless of the location of the source node and the group of corresponding receivers. Similarly, the group leader that is chosen to be the “core” router of a delivery tree also remains the same irrespective of several network dynamics such as node mobility and group membership. A single delivery tree with shortest paths to receivers does not necessarily guarantee minimum overall end-to-end multicast delay [41]. This can be attributed to several factors such as the variabilities in the wireless medium, queueing delays, unfair MAC scheduling, and variable contention that results in reduced bandwidth especially at the tree replication points. Thus, from the perspective of the multicast application, a single delivery tree with the same node fixed as a core is not amenable to adaptations to MANET dynamics such that the overall delay experienced by all receivers is either limited to some desired QoS threshold and/or are equal. Hence, the shared tree protocol has to be modified to make it Quality of Service (QoS) sensitive.

QoS can be implemented in a number of ways and the complexity of the solution depends upon the amount of network state maintained, exchanged, and manipulated among the multicast members [13] [14]. Also, on a higher level of network abstraction, QoS provisioning [23] [44] [27] can be viewed as some form of resource reservation [46] with multicast members participating in resource

signalling [22] [5] [3], organization [24] [33] [36], and management [45] [1]. For a tree-based protocol, tree reconfiguration is necessary but not sufficient for any QoS implementation for MANETs. QoS protocols for MANETs are therefore heuristics that actively pursue some form of local and/or global tree reconfiguration. An example of a global tree reconfiguration could be the reorganization at each hop of the tree to manipulate the child memberships of the parents. Such a scheme would be overkill due to its large overhead and its immediate possibility of becoming sub-optimal with changing network dynamics. We therefore propose a localized reconfiguration scheme that decides upon the optimality of the position of the existing tree group leader and its 1-hop neighbors. In this paper, we consider only multicast application delay as our metric to perform core selection.

For a given multicast tree topology, the delay monotonically increases as the message travels greater distances. Our algorithm monitors the delay on neighboring branches by way of receiver acknowledgments to respective multicast packets and it also maintains a history of these delay samples. The core then estimates the delay over some number of delay samples to find a neighbor that experiences the highest branch delay. The core selects this neighbor as the new core only if the migration to this neighbor will not increase the delay estimated on other branches above a pre-specified QoS threshold. With changes in network topology the core moves toward a currently optimal node. Thus, the core is always moving toward a currently optimal position such that the delay is equalized from all its subtree branches and/or these delays are restricted to an application-specified QoS threshold.

In cases where the source is not the tree-core, the core-migration approach ignores the possibility of a reduced unicast route length from the source to the tree-core that may further minimize the overall multicast application delay. We consider only the forward delay from the tree core to the multicast members. Multicast receivers include this delay in their acknowledgments by calculating the relative time difference between the transmission of the packet (included in multicast packet headers as a separate *timestamp* option by the core) and its corresponding reception. The acknowledgments are then forwarded along the reverse shortest path back to the core with nodes in the intermediate path forwarding only the maximum delay heard for that sequence number. It can be clearly seen that this delay is not an accurate representation of the topological organization of the tree. On the contrary, the delay measured by the core is actually all inclusive of the wireless transmission delays, path-length delays, queuing delays caused by congestion, and retransmissions due to packet collisions. We assume however that path length delays contribute significantly to the overall delay monitored at the core. Also, since the core migration decision is based on a history of delay samples, transient effects on delay calculations by congestion and collisions can be neglected without significant loss in the efficiency of the algorithm.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 focuses on the design philosophy of our QoS-aware core migration protocol. We present simulations in section 4 that compare the QoS efficiency of multicast on a tree-based multicast protocol such as MAODV with and without the QoS-aware core migration technique. Finally, section 5 presents the conclusions and discusses future work.

2 Related Work

The problem of QoS signalling, provisioning, and management has been a hot topic of research in both wired and wireless networks. In contrast to fixed networks, QoS support is very challenging in ad hoc wireless networks where nodes have an additional degree of freedom by way of uncontrollable and unpredictable mobility. However, QoS solutions usually involve measuring a certain quality of the network at different time instants in order to gain an insight into its respective performance characteristic(s). Wired networks are very flexible to such characterization both at the centralized and distributed level. However for ad hoc networks, even distributed measurement of network performance is not enough as such a measurement is unsteady due to the unpredictable nature of the wireless medium. Moreover, mobility makes cooperative participation for measurement and exchange of quality metrics very difficult among wireless nodes.

QoS solutions for wired networks are therefore insufficient for MANETs. For example, the Integrated services (IntServ) [5] and Differentiated services (DiffServ) [3] architectures that are popular on the Internet are not applicable for MANETs. Our proposal differs from both IntServ and DiffServ methodologies. First, we do not provide QoS reservation on a per-hop or per-flow basis. Instead, we provide QoS for every multicast group on an end-to-end basis that incorporates tree branches connecting the core to all the multicast receivers. Initially, the source can be selected as the core for the multicast tree. However, with continual degradations in the perceived end-to-end QoS (i.e. delay), the core migrates gradually to any of the neighboring on-tree routers that exhibits such degradations. Without using an explicit signalling protocol such as RSVP [46] that reserves per-flow resources, our solution provides end-to-end delay guarantees. However, we only provide QoS on a best-effort basis as opposed to IntServ that additionally provides two more service classes such as guaranteed and controlled load.

There are a number of proposals in wired networks under the general area of distributed center location algorithms [37] [17] that loosely resemble our QoS solution for MANETs. Usually these algorithms involve selecting a subset of best nodes from the whole network by minimizing a certain weight

function such as average delay, average distance, maximum network diameter, and minimum hopcount. Also, their migration approach frequently transcends a number of hops whereas our protocol sparingly but assuredly migrates on a 1-hop basis with gradual and systematic adaptations to MANET dynamics. Thus, these center selection methods for wired networks are all not feasible for the MANET environment as they require complete participation among all the network nodes and almost total information about the network topology.

In core-based tree (CBT) multicast protocols, the shortest path from a requesting router to the core may not be the best QoS route. There are a number of reasons for this problem, with the most influential factor being the organization of the tree that results in the concentration of traffic near the core and the various tree branch points thereby exhausting network resources. Several approaches [6] [7] [2] based on a bidding process have been proposed that usually involve a requesting router performing a TTL-scoped flooding to identify a path to an on-tree router based on certain QoS metrics. Although these protocols are QoS sensitive, they do not address how *end-to-end* QoS is achieved; they focus only on finding the best route (or the best on-tree-router) along which (at which) a new member joins the tree. Moreover these approaches have been developed specifically for wired networks such as the internet and they are not suitable for MANET dynamics.

QoS protocols for MANETs depend upon a number of architectural choices such as a flat or hierarchical organization, proactive or reactive (on-demand) routing protocol, performance metrics, topological update strategies, and size of network participation. Moreover, there are also a variety of layer-specific and cross-layer QoS solutions for MANETs. However, our core migration approach for providing QoS is specifically a network layer solution for a tree-based organization that allows multicast group communication services. We expect the application layer to describe the service requirements in terms of the desired delay ($delay_{target}$) and its $delay_{threshold}$ which is a margin above the $delay_{target}$ that the multicast application can safely tolerate. An interesting discussion of various approaches to QoS provisioning for group communication in MANETs has been furnished in [27]. We focus primarily on those protocols that consider an additive delay metric as one of their routing constraints. We also discuss research efforts that propose a novel technique for performance calculation using a certain history of sample measurements. For the sake of comparison, we also discuss approaches that pursue incremental and hop-by-hop discovery of neighbors to form an end-to-end QoS-aware routing path.

The Core Extraction Distributed Ad hoc Routing (CEDAR) [36] protocol dynamically selects a minimum set of dominating nodes as the core of the network. QoS routing is achieved by propagating the bandwidth availability information of stable links to all core nodes. Thus, by having only a few core nodes, an efficient and low overhead link-state maintenance infrastruc-

ture is established. QoS route computation is done by the core nodes using only locally available bandwidth information. The only similarity between our approach and CEDAR is the use of core(s) to maintain QoS information. However in our case, the delay perceived by the multicast receivers at the network edge is gathered and maintained incrementally at the core node. Although, the route computation and tree maintenance is done by the underlying multicast routing protocol, core migration based on the end-to-end delay metric results in a QoS-aware routing tree. Also, since the acknowledgments that carry the delay information are *compounded* at every parent and are forwarded to only *one* core, the overhead involved in state collection and maintenance is significantly lower compared to CEDAR.

As mentioned earlier, MANETs are impractical for precise measurement and collection of network performance metrics. Imprecise measurements are therefore used to approximately perceive any enhancement/degradation of desired service qualities. Chen and Nahrstedt [10] propose a ticket-based probing algorithm to limit the flooding that is used to discover a QoS-aware routing path. Each probing message containing at least one ticket is split into multiple probes and is flooded to next hop neighbors to find a QoS-aware path to the intended destination. When one or more probe(s) arrive(s) at the destination, the hop-by-hop path along with its delay/bandwidth information is known. This information can be used to perform resource reservation for a path that satisfies the desired QoS. Since probabilistic schemes are not suitable in MANETs for calculating path-representative delay values, the authors propose a smoothing formula to estimate the current delay from a history of delays. Their estimation is represented as a range of $[delay - \delta, delay + \delta]$, where δ is the estimated maximum change in delay before the next update of the delay history. The objective of our weighing scheme is to give appropriate importance to both recent and past delay samples in the history. In other words, using the concept of mobility prediction and adaptively changing the weighing factor one can give importance to latest samples for high mobility situations and average all the samples in the history for moderate to low mobility scenarios. However, this paper deals with a static weighing scheme. We are still exploring the advantages of an adaptive scheme.

With a layered approach to protocol interactions, performance parameters at different layers need to be shared across layers for providing a coherent view of the state of the network. Cross-layer protocol optimizations for providing QoS in MANETs have therefore become a necessity [12]. A middleware that provides dynamic QoS ranges and allows adaptation in application behavior according to the feedback of the lower layers is desirable. However, our proposal assumes a static specification of QoS requirements from the user. Also our protocol needs tree-based infrastructure support from the multicast routing protocol at the network layer. There are obvious benefits to using cross-layer optimizations that are arbitrated by the middleware [23] [9] and

we consider this to be future work.

3 Design Philosophy

The following subsections provide an in-depth analysis of the characteristics of tree-based multicast as it applies to our core migration algorithm for providing QoS in MANETs. We provide detailed explanations for our proposed delay monitoring scheme. An accumulation of delays for multicast packets over time results in a history. We explain how this history of delay samples represents the QoS snapshot of a tree-based network organization at various time intervals. We also discuss a novel scheme of recursively weighing certain samples of the delay history to get an estimate that gives appropriate importance to past and recent delays. This *delay estimate* ($delay_{estimate}$) in conjunction with application specified QoS parameters such as *delay target* ($delay_{target}$) and *delay threshold* ($delay_{threshold}$) is used by the core to calculate the *QoS imbalance* of the tree branches. We provide two optimizations that affect the core migration decision and its migration frequency as applicable to MANET dynamics. We also discuss relevant features that prevent migration oscillation and reduce migration overhead. In the following discussion, we will use the terms *core* and *group-leader* interchangeably.

3.1 Network Model

A mobile ad hoc network can be modeled as a network consisting of n identical mobile hosts (nodes). These mobile hosts employ a packet radio network to communicate with each other. Multiple nodes falling in the radio coverage area of another node are said to be neighbors of this node and they can simultaneously receive a message transmitted from that node. The topology of the mobile ad hoc network is dynamic. Such a network can be represented by an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of logical links between neighboring nodes. We assume that nodes leave and join the multicast group arbitrarily. We also assume that the link layer protocol takes care of transient link faults and that all failures in the system are temporary. Nodes that fail permanently can be removed from the multicast group, but we do not consider that part of the multicast protocol in this paper.

3.2 *Tree-based Multicast*

As mentioned earlier, our core migration protocol is applicable only for a tree-based multicast in which most of the traffic flows through a specific node known as the core. Figure 3(a) shows a tree where the node at the top is the core, nodes at the bottom are leaves and the intermediate nodes act as parents to their downstream neighbors whereas they act as children for their upstream neighbor. This tree organization is maintained in the presence of node mobility. The core migration approach therefore needs a multicast routing protocol such as MAODV that maintains such an organization for MANETs.

The core sends data on its outgoing interfaces and the intermediate nodes forward it to their downstream neighbors. The multicast data eventually reaches the leaves after which it cannot be forwarded any further. The core timestamps every multicast packet and intermediate nodes on reception calculate the forward or downstream delay as the difference in the transmission timestamp and reception time. The leaves however generate a reverse acknowledgement that is sent upstream toward the core. Thus in figure 3(b), nodes 5, 9, 12, 13, 14, 15, 16, 18, 20, 21, 23, and 24 will generate acknowledgements whenever they receive multicast packets. A leaf node calculates the downstream delay it experienced for respective data sequence number and updates it in the acknowledgment message. As this acknowledgment traverses upstream, intermediate nodes update this packet with the maximum delay that was heard so far for that sequence number from any of its children. A parent can either wait for acknowledgments for the same sequence number from all its children and then decide upon the maximum delay or it can just update and forward the maximum delay heard so far with reception of every acknowledgment. For mobile ad hoc networks, with dynamic link breaks and joins, waiting for acknowledgements for every child seems impractical. Also since the topology is dynamic, prompt reception of approximate delay values for a branch conserves the liveness of that delay sample. The more the wait, the more the probability of the delay value becoming stale.

In order to conserve bandwidth, an acknowledgement message includes delay samples for some fixed number of data packets (say, MAX-QoS-SAMPLES = 5). Thus, a leaf will generate an acknowledgement only when it has the 5 latest unacknowledged delay values. For every such 5 data sequence numbers, the leaf updates the downstream delay it experienced. Intermediate nodes that have information about these samples will either update the corresponding information in the acknowledgement packet or their local delay values for those sequence numbers. Again, this decision is based on previous knowledge about delay heard so far for this packet.

On the reception of acknowledgment packets from its neighboring on-tree

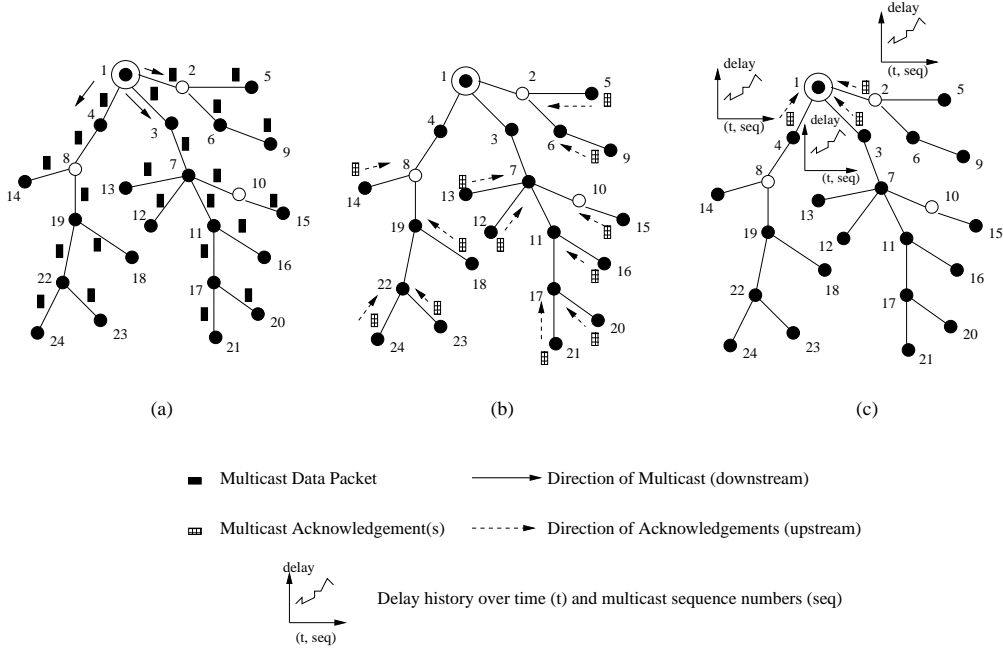


Fig. 3. QoS-aware Multicast Tree: (a) Core initiating multicast, (b) Tree leaves initiating reverse acknowledgements, and (c) Core receiving compounded acknowledgements and constructing a delay history for its outgoing subtree branches.

routers (OTRs), the core builds a history of delay samples for the corresponding tree branch. Thus from figure 3(c), the core builds delay histories as it receives acknowledgement packets from OTRs 2, 3, and 4. The core attributes the history of downstream delays for the respective subtree branches to its neighboring OTRs. Thus, by forwarding recently known maximum delay values and suppressing old sequence numbers with lower delay values on a periodic basis, the core receives a fairly consistent view of the QoS characteristics of the underlying topology. The core uses these delay histories to determine that some tree branch is consistently showing maximum delay. The core topologically migrates one hop nearer to the leaves of such a branch in the hope that it will eventually reduce the path length and the corresponding downstream delay. The core informs all the multicast members of its decision to migrate by multicasting a *new-core-update* message.

The amount of state maintained in terms of the number of delay values stored at each node is minimal. A percentage of this state (say, $purge_{factor} = 75\%$) gets purged automatically with reception of every *new-core-update* message. In case of tree partitions, we have a core for every partition and each core tries to perform QoS-aware migration for its respective partition. The underlying multicast routing protocol such as MAODV usually merges two partitions whenever they discover each other in their vicinity. In such situations, the core with the lower IP address initiates tree repair thereby avoiding loops. Support for reliable message delivery by the multicast protocol will enhance

the effectiveness of the core migration algorithm due to the availability of delay samples on a continual basis. However, if some samples are lost, the protocol waits for new samples to construct the history.

3.3 Delay Estimate

The decision for core migration is obtained by calculating the $delay_{estimate}$ over the history of delays gathered by the current core over each link to which the core is connected. Figure 4 depicts a possible history of delays accumulated by the core from the three OTRs 2, 3, and 4 (see figure 3). The core stores the samples of delay obtained from each of the m neighboring on-tree routers as a history in an array $delay_{OTR_j}[i]$, where $1 \leq j \leq m$ and $0 \leq i < \text{MAX-QoS-SAMPLES}$.

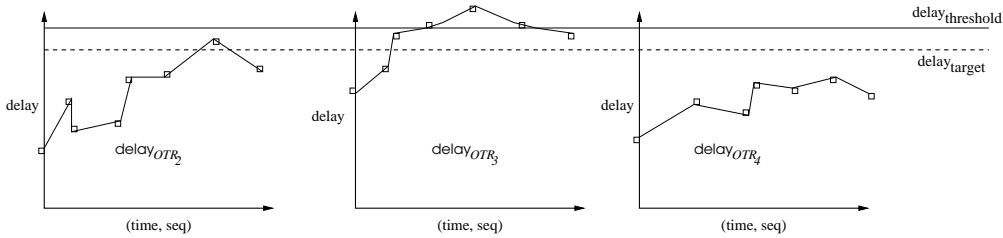


Fig. 4. History of delays used for core migration

In order to decide which branch is showing a consistently high delay, the core has to periodically compute the $delay_{estimate}$ over some fixed number of samples (MAX-QoS-SAMPLES) for each of the m neighboring on-tree routers. The recursive equation given below computes $delay_{estimate}$ as a weighted average thereby giving the desired importance to past and recent delay samples.

$$delay_{estimate_j}[i] = \alpha delay_{estimate_j}[i - 1] + (1 - \alpha) delay_{OTR_j}[i] \quad (1)$$

That is, the $delay_{estimate}$ at step i is $\alpha \times delay_{estimate}$ at step $(i - 1)$ and the $delay_{OTR_j}$ at step i , where $0 \leq \alpha \leq 1$, $0 \leq i \leq \text{MAX-QoS-SAMPLES}$, and $1 \leq j \leq m$.

A small value of α , say $\alpha = 0.2$, gives more weightage to recent delay values whereas $\alpha > 0.5$ gives more importance to past values. Since a mobile ad hoc network is characterized by dynamic changes in network topology, assigning importance to the most recently monitored values enables the core placement protocol to adapt to the most recent changes in network topology without neglecting the remaining history of delays monitored by the core. For example, $\alpha = 0.5$ gives equal significance to both the current and past samples.

Intuitively, the value of the $delay_{estimate}$ allows the core to locally perceive a comprehensive global view of the imbalances in delay over its subtree branches purely on the basis of the history monitored by it. As mentioned earlier, one of the major contributions to these imbalances in delay is due to the frequent asymmetric changes in the network topology. In other words, $delay_{estimate}$ summarizes these changes of network topology over some monitoring period (equivalent to monitoring *MAX-QoS-SAMPLES*) and is therefore an important parameter in our core migration algorithm. In the subsequent section, we will be using this $delay_{estimate}$ value along with $delay_{threshold}$ to move the core to the next hop in an attempt to minimize these delay imbalances. Now, referring to figure 4, it can be concluded that the OTR_3 delay graph shows a *large and persistent deviation*. That is, for OTR_3 , it is possible to have $delay_{estimate} > (delay_{target} + delay_{threshold})$. Hence, OTR_3 would be the next probable position for the core.

3.4 Core Migration

The selection of some node as a core for a multicast tree such that its tree branches form shortest paths to group receivers is not enough to minimize end-to-end multicast delay. This is because the nature of the wireless medium is unpredictable. Also, traffic concentration at the core and at tree branch points results in delays that are independent of the shortest path tree organization. Intuitively, if the knowledge of the complete network topology was available, one could propose a solution that meets two diverging requirements of simultaneously constructing shortest-path trees and also performing per-hop tree reconfiguration. However, any such solution that aims to cumulatively meet two or more concave or additive constraints in any possible combination has already been proved to be NP-complete by Wang and Crowcroft [42]. Assuming that an approximate network organization that loosely meets our design goals was possible, it would repeatedly become suboptimal by random node mobility in MANETs. The frequent (re)construction of such an organization would thus be prohibitively expensive for MANETs.

Our core migration heuristic identifies these limitations. As discussed earlier, monitoring the history of delays gives the core an approximate snapshot of the performance of the tree. This delay is all inclusive of the path delay, contention delay, MAC delay, and queueing delay. Since tree reconfiguration is not possible at every hop, we propose changing the organization of the tree one hop around the core. That is, we propose migrating the core to a tree branch that has been consistently experiencing higher delays. In this way, the core moves topologically closer to the end receivers (or leaves) on that tree branch thereby reducing the downstream delay experienced by these nodes. However, such a migration toward a neighboring tree branch would result in

increased delay in other branches. We take advantage of application specified QoS parameters such as $delay_{target}$ and $delay_{threshold}$ in order to make a better core migration decision.

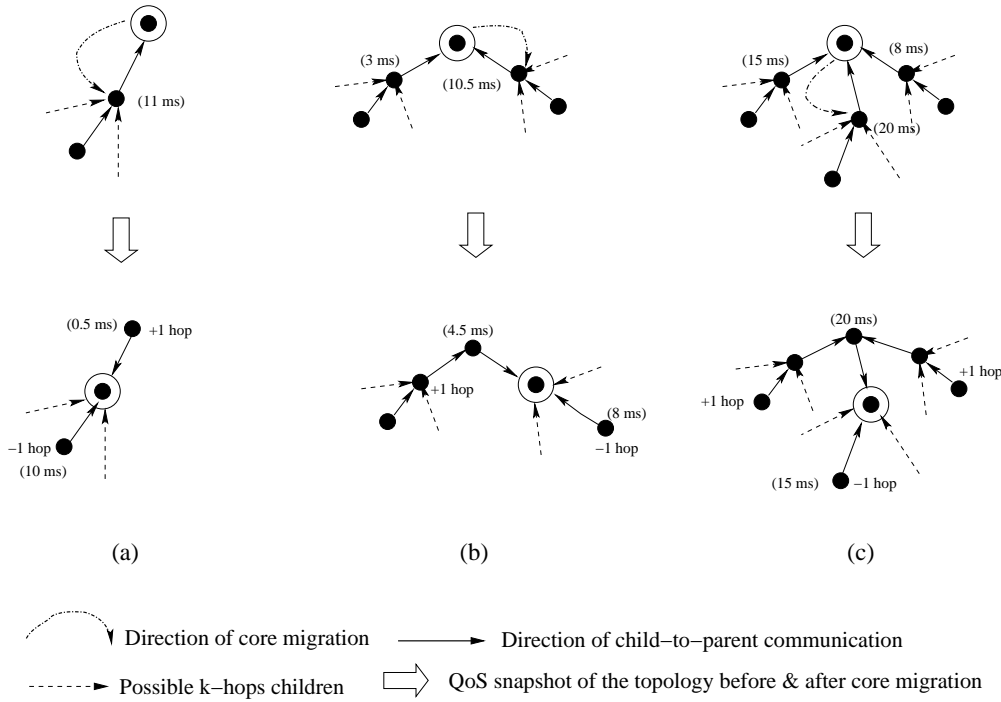


Fig. 5. Core migration scenarios for a core with variable $delay_{estimate}(s)$ and 1-hop neighbors: (a) 1-otr, (b) 2-otrs, and (c) 3-otrs.

In figure 4, we define $delay_{target}$ as the desired QoS specified to the core by the multicast application and $delay_{threshold}$ as the maximum limit above the required QoS value that the multicast application can safely tolerate. The $delay_{threshold}$ is essentially used to avoid frequent core migrations due to *miniscule* changes above the $delay_{target}$. The value of $delay_{threshold}$ is dependent upon the minimum delay improvement that could occur with 1-hop core migration. Apparently, this value depends primarily upon the dynamics of the mobile ad hoc network and secondarily upon the tolerance in service quality acceptable by the multicast application. Figure 5 shows core migration scenarios among its one-hop neighbors. For all the three scenarios, let us assume that the $delay_{target} = 8$ ms and $delay_{threshold} = 2$ ms. The rule for migrating the core to the next hop is that the delay as perceived by the core on that hop should be the maximum of all delays and that this delay should be either greater than or equal to the sum of $delay_{target}$ and $delay_{threshold}$.

With scenario 5(a), the core has only one downstream neighbor and so any migration decision would involve that neighbor. For this particular scenario, the core perceives a $delay_{estimate}$ of 11 ms and after migration, the new core meets the application-specified requirements from both of its new neighbors. A

similar concept applies to the core migration scenario with two 1-hop on-tree routers (see figure 5(b)). It should be noted here that with each migration, the core moves topologically closer by a hop on a branch where it migrates, whereas it moves one hop away from all the other neighboring branches. The scenario in figure 5(c) is a special case, where the core cannot meet QoS requirements on all but one branch that has been experiencing a satisfactory delay of 8 ms. One of the secondary objectives of core migration is to balance the delays on all its downstream branches. This happens specifically in situations where it cannot meet its primary objective of minimizing delays across all its branches to within the $delay_{threshold}$. Therefore the core migrates to a branch with the maximum delay. Intuitively, such a solution would lead to an oscillating state where the core migrates back-and-forth among its neighbors.

To reduce unnecessary overhead caused by repetitive migration, we propose an optimization that delays the migration decision for an appreciable time until the network has stabilized. It is clear that with mobility, the topology of the network will change such that new link breaks and joins will result in tree branches with different delay scenarios. The availability of a new history of branch delays may provide optimistic opportunities for core migration such that the delay is kept to within the $delay_{threshold}$ limits or are balanced. Thus this core migration optimization, discussed in the next section, uses mobility and randomization in MANETs along with the time to their best advantage.

3.5 Core Migration Optimization

With reception of acknowledgments from neighboring on-tree routers (OTRs), the core constructs a delay history for its subtree branches over time. Leaf nodes that are closer to the core in terms of hop counts usually acknowledge a data packet much earlier than the ones that are farther away. The decision to migrate can then be made either on a per-acknowledgement basis (that includes MAX-QoS-SAMPLES) or on a per-OTR basis. In the first case, the core migrates with every reception of acknowledgement packet from any OTR, whereas in the second case the core waits for acknowledgments from all its neighboring OTRs to make a decision. Thus, although in the first case the core may make a decision on recently received samples, its migration is still biased to shorter sub-tree branches than longer ones. However, in the second case, the core due to its obligation to wait for acknowledgements from all OTRs would have to wait long. Thus in this case the decision may be biased by old history samples.

The purposes of postponing the migration decision are thus threefold. On one hand it reduces oscillation among neighboring OTRs, especially those that return delays within each other's range. With an increase in wait times for

migration decisions, shorter tree branches have less opportunity to dominate the migration process. Third, it allows a decision to occur not only over a larger history but also over a number of neighboring OTRs.

3.5.1 *2-otr optimization*

We first provide a simple solution that allows migration decisions over a history that is accumulated from a minimum of one acknowledgement packet (i.e. *MAX-QoS-SAMPLES*) each from at least two OTRs. We call this solution as the *2-otr optimization*, as it optimizes the migration decision to any of the two OTRs having recent delay histories. In a way, this optimization forces the core to consider a minimum of two neighbors although the total number of neighbors it might have may be more or less than two. Thus, by considering recent samples from only two neighbors, this optimization consistently allows the migration decision to adapt to network dynamics over shorter durations.

3.5.2 *Count Acknowledgements and OTRs*

In this solution, the core decrements the number of OTRs it waits with reception of each acknowledgement packet. We introduce another parameter known as *MAX-ACK-PACKETS* that the core waits for before making any migration decision. For example, if a core had 4 neighboring OTRs and if the protocol had the value of *MAX-ACK-PACKETS* = 5, it will either wait for a total of 5 ack packets from any OTRs or one ack packet from each of its 4 neighboring OTRs. In order to allow flexibility, the core decrements the number of OTRs it waits for, with the reception of acknowledgement packets. So, if one sub-tree branch was not able to report any values, the core would give it a time equivalent to receiving 5 acknowledgement packets before making any migration decision.

This solution minimizes the chances of migration oscillation by providing a delay proportional to the time required to wait for acknowledgements from some or all of the neighboring OTRs.

4 Analysis of Protocol Performance and Correctness

In this section, we present proofs of correctness that the protocol gradually moves the core toward the optimal location and reaches that location when the network is stable. Further, simulations are also conducted to analyze the performance of MAODV multicast with and without our core migration extension. We also study the effects of the weighing factor α on the QoS-efficiency

of our protocol under changing network dynamics.

4.1 Correctness

Theorem 1 *When the network topology is static, the core migrates to an optimal router.*

Proof. The core monitors the delay on the subtree branches by way of the local time at which multicast packets are sent and the corresponding acknowledgments received. For a given multicast tree topology, the delay monotonically increases as the message travels greater distances. Moving the core by one-hop on a branch with the highest delay decreases the delay on that branch. However, the delay correspondingly increases on the other branches. It should be noted here that the core only migrates to a branch whose delay is not only the highest but also above the threshold specified as QoS by the multicast application. In situations, where the delay in all branches are above their thresholds, then core migration aims to reach an optimal position where it topologically balances delays across all its branches. Conversely, when delays across most of the tree branches are less than the threshold, then migration by one-hop on a branch with delay higher than the threshold is only going to increase delays on other branches by at most a threshold magnitude. The choice of threshold as given by the multicast application is much greater than one-hop transmission delay. Since the network topology is static by assumption, the core continuously migrates on a hop-by-hop basis until reaching a position where the differences in delay are not above the *threshold* necessary to migrate. Thus, the core eventually migrates to that optimal position where the delays from all the subtree branches are approximately equal, which is at the center of the multicast tree. \square

Theorem 2 *When the network topology is changing, the core moves toward an optimal router for the current topology.*

Proof. When the network topology changes, the $delay_{estimate}$ that is computed for the n samples also changes accordingly. Thus, hop-by-hop migration of the core leads the core toward the best position for the current topology. At steady state, the core would migrate to an optimal position where it equalizes the delay coming from all the subtree branches. Since the topology is changing, the core moves toward the currently optimal core. The migration may change toward a different optimal node when the topology changes dictate a new optimal point. Thus, the core is always moving toward a currently optimal position. \square

4.2 Simulations

The simulations have been performed using ns2 (version 2.26) [39] as our discrete event simulator. Each mobile node on flat ground uses an omnidirectional antenna with gains equal to 1 (G_t or $G_r = 1$), with the antenna height at 1.5m (h_t or $h_r = 1.5$). The wireless interface works like the 914MHz Lucent WaveLAN DSSS radio interface [38], with the system loss $L = 1$. The power of the signal attenuates based on the free space model at short distances and the two-ray ground reflection model at longer distances. The crossover point is around 86.4m. The transmitted signal power is around 0.2818W and the correctly received signal threshold is around 3.652e-10W, so the transmission range is about 250m. The MAC layer protocol used in the simulation is the IEEE 802.11 [11] Distributed Coordinated Function (DCF) with unslotted CSMA/CA used to sense an idle wireless channel and avoid collisions. The bit rate is 2Mbps.

Nodes move on flat ground and the movement is modeled by the random way-point model [20]. Each node moves from a random location to a random destination with a randomly chosen speed uniformly distributed between 0 and a given max speed. When the destination is reached, after pausing for a given pause interval, another movement with random speed and direction begins. This behavior repeats for the duration of the simulation. Thus, node mobility scenarios can be characterized by maximum speed and pause time. Continuous movement is equivalent to 0 pause time, while no movement is obtained when the pause time is set to the duration of the simulation. The units for pause times is seconds and for maximum speed is meters/second (m/s).

We are using an MAODV implementation with an added proactive tree connection maintenance feature available from Carleton University [47]. The basic idea of their proposal is to predict tree link breakages and proactively construct alternative routes before they become unavailable. AODV is our default unicast routing protocol in the simulations. To exclude any influence by the unicast traffic, every node in ns2 maintains a network interface that includes two send buffers. One for queuing unicast packets and the other for multicast data packets. Both queues are FIFO buffers with a maximum size of 64 packets and they retain packets for less than 30 seconds.

We use a constant bit rate (CBR) data traffic pattern with fixed size data packets of 256 bytes that are sent out roughly at the same time interval. The core migration extension adds a timestamp field to every multicast data packet which is of two bytes. We also add a new acknowledgement message that is generated by the leaves and forwarded upstream toward the core. Every acknowledgement message carries updated delay information for a fixed number

of data packets (*MAX-QoS-SAMPLES*) that are identified by their unique sequence numbers. At each tree branch point, a parent aggregates the acknowledgements for the sequence numbers that it has heard so far from its downstream neighbors or children. Also, in addition to the *source id* we also include the *id* of the node forwarding the acknowledgement message. The core thus attributes the history of its subtree branches to its neighboring OTRs (i.e. *id* of the nearest 1-hop ack forwarders).

We use a rectangular fields of 1500m×800m, 2000m×800m, 3000m×800m as our simulation space instead of a square one. The idea was to have MAODV form longer routes for performing multicast over a tree. However, since the height of the tree from its group-leader (or core) to its leaves depends upon the position of the group leader which is always migrating, we may have shorter or longer routes at different simulation times. Moreover, a deployment area with one of its dimensions comparable to the node transmission range of 250m would not result in an appreciable change of routes/links.

There are 50 nodes in the simulation. All the multicast receivers are group members whereas the multicast sender(s) are non-members. We conduct simulations for 1 sender^a with the number of receivers being 40. All the members join the group at the beginning of the simulation and they retain their membership till the end. A simulation time of around 30 seconds is allocated for multicast members to form an initial group tree. A multicast sender then begins transmitting its data packets at a rate of 2 packets/s. After 2000 seconds the multicast source stops transmitting the data traffic and the simulation ends. We conduct simulations for both core migration optimizations. The weighing factor α is set to 0.8. Also, the value of *MAX-ACK-PACKETS* is set to 3 for *count-acknowledgements-and-otrs* optimization. For the sake of comparison, we commented out our core migration code within MAODV and performed the simulation without our proposed QoS-aware extensions. We call this simulation the static core scenario. For each simulation area we generate 7 position scenarios with nodes randomly starting from different positions. We also generate 20 different mobility patterns for every position scenario. Thus, we have 140 possible scenarios for each area of deployment. For all these scenarios, we set pause time to 0 second and maximum speed set to 20 m/s. The $delay_{target}$ is set to 40ms and $delay_{threshold}$ is set to 5ms.

^a MAODV constructs a separate multicast group for every sender. Hence, simulation for multiple senders becomes redundant as it would amount to performing core migrations for every individual group with a single sender.

4.3 Simulation Results

In order to highlight the performance obtained with core migration, we account for the maximum delay experienced for every packet received by any multicast member. These packets are then grouped into 8 different QoS scenarios from 15 ms to 85 ms. For simplicity, we include the percentage packets received for every QoS scenario. Then we average this packet percentage value (\bar{p}) across all the 20 mobility scenarios for each position scenario. Tables 1–3 reflect the average best and worst case QoS statistics selected from the 7 position scenarios along with their standard deviations (σ).

With increase in the area of deployment, the network becomes sparse. The reverse applies when the area of deployment is reduced. Since the wireless radios have a fixed transmission range of 250m, a dense deployment would not offer reduced delay performance as compared to a moderately sparse deployment. This can be attributed to frequent collisions which is due to the pronounced hidden and exposed terminal effects. Also since network routes do not change appreciably in a dense deployment, providing QoS with frequent core migrations in such a deployment seems futile. We have verified this observation by performing simulations for smaller deployment areas such as 1500m×300m, 1000m×300m, and 800m×300m. As we go to the other end of the deployment spectrum, we find that with increase in network area, the network becomes sparse and there is an appreciable degradation in network connectivity. We have therefore chosen deployment areas 1500m×800m, 2000m×800m, and 3000m×800m that are moderately sparse. These deployment areas provide an insight into the QoS-provisioning efficiency of core migrations in a tree-based multicast organization.

Table 1
QoS stats with no core migration (static core)

QoS	1500m×800m				2000m×800m				3000m×800m			
	Best		Worst		Best		Worst		Best		Worst	
	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ
(0.010, 0.005)	5.64	1.76	5.07	3.76	7.25	2.35	5.73	3.34	10.23	5.49	9.12	7.61
(0.020, 0.005)	11.78	2.83	10.40	5.58	15.67	4.79	13.33	6.90	21.94	5.94	18.70	9.48
(0.030, 0.005)	18.54	3.87	16.78	6.25	25.76	7.17	22.79	9.23	38.69	6.40	36.98	9.37
(0.040, 0.005)	26.12	4.91	24.10	6.66	36.60	7.88	33.07	10.00	48.59	5.70	45.56	9.69
(0.050, 0.005)	34.84	5.72	32.37	7.07	47.80	7.69	44.29	8.95	66.97	5.25	63.84	9.40
(0.060, 0.005)	45.74	5.40	42.76	7.09	59.71	6.44	56.02	8.40	76.27	4.84	74.33	7.34
(0.070, 0.005)	57.82	5.51	55.08	7.38	70.24	4.87	67.27	6.68	81.14	3.64	77.44	5.21
(0.080, 0.005)	69.84	4.97	67.37	7.14	79.19	3.59	76.92	5.16	88.55	2.61	86.10	4.54

Table 2
QoS stats for core migration with 2-otr optimization

QoS	1500m×800m				2000m×800m				3000m×800m			
	Best		Worst		Best		Worst		Best		Worst	
	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ
(0.010, 0.005)	5.91	1.48	4.53	3.81	9.07	1.77	6.31	5.09	17.55	4.15	15.01	6.73
(0.020, 0.005)	13.10	2.91	10.84	5.47	22.91	4.33	15.37	8.59	34.89	5.30	31.20	9.23
(0.030, 0.005)	21.28	4.06	18.64	6.28	36.63	6.71	25.44	10.02	47.98	5.35	43.63	9.64
(0.040, 0.005)	32.68	4.80	27.65	6.24	48.72	6.44	36.38	11.21	65.03	5.21	61.14	9.10
(0.050, 0.005)	45.39	4.02	38.20	7.47	59.66	7.67	48.53	8.37	78.96	5.12	74.05	8.23
(0.060, 0.005)	56.39	3.55	50.08	8.40	69.62	6.61	59.72	8.51	86.27	3.61	80.99	7.18
(0.070, 0.005)	67.25	3.88	62.22	8.11	77.64	4.98	70.93	7.30	92.99	2.65	88.54	4.09
(0.080, 0.005)	75.81	4.06	73.24	8.69	84.25	3.38	78.48	6.69	96.13	2.21	93.45	2.56

Table 3
QoS stats for core migration with count-qos-packets optimization

QoS	1500m×800m				2000m×800m				3000m×800m			
	Best		Worst		Best		Worst		Best		Worst	
	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ	\bar{p}	σ
(0.010, 0.005)	7.32	1.50	5.60	3.07	9.17	2.04	5.93	9.66	13.79	4.77	10.68	7.03
(0.020, 0.005)	16.71	2.73	11.43	5.17	21.07	3.98	13.96	6.81	29.74	6.21	25.36	9.39
(0.030, 0.005)	25.12	3.70	21.03	6.39	34.03	4.69	22.96	8.69	45.69	6.32	40.10	9.43
(0.040, 0.005)	34.37	4.35	30.33	6.85	46.87	5.14	34.39	9.49	60.25	6.16	54.66	9.72
(0.050, 0.005)	44.54	4.37	38.15	6.39	57.76	5.57	47.70	8.11	72.12	5.05	66.63	8.51
(0.060, 0.005)	57.46	4.02	50.03	7.91	67.78	4.57	59.70	8.82	80.61	4.53	76.39	7.26
(0.070, 0.005)	68.12	3.82	62.36	8.09	75.92	3.76	70.04	6.79	86.41	4.32	84.03	5.25
(0.080, 0.005)	76.33	3.88	73.66	8.11	82.93	3.42	78.32	5.34	90.92	2.82	89.12	4.07

From tables 1–3, we can make the following general observations:

- (1) Less number of packets meet minimum QoS delay margins. Conversely, increasing percentage of packets can meet a relaxed delay requirement.
- (2) With increase in the length dimension of the area of deployment, nodes get a higher degree of mobility freedom. In this case, although the routes are longer, they offer better QoS-delay performance as opposed to shorter routes in relatively dense deployment.
- (3) The standard deviation usually becomes larger with increasing network deployment area. This can be attributed to a higher chance of network partitioning in sparse deployment areas.
- (4) Tables 2 and 3 show a marked improvement in QoS performance in comparison to table 1 which is the performance without any core migration.
- (5) Comparing table 2 with 1, we find that the improvement is better in a deployment area of 3000m×800m whereas the improvements are very modest in a 1500m×800m area.

- (6) Comparing table 3 with 1, we find that the improvement is better in a deployment area of 1500m×800m whereas the improvements are very modest in a 3000m×800m area.
- (7) The 2-otr and the count-qos-packets optimizations perform relatively similar for a 2000m×800m deployment area.

We have performed simulations with the nodes randomly moving according to the random-way-point model. In this model, nodes bounce which by itself is not a good representative model of the ideal world.

Table 4

Migration frequency for both the core migration optimization schemes

Best/Worst	2-otr optimization						count-qos-packets optimization					
	1500m×800m		2000m×800m		3000m×800m		1500m×800m		2000m×800m		3000m×800m	
	\bar{m}	σ	\bar{m}	σ	\bar{m}	σ	\bar{m}	σ	\bar{m}	σ	\bar{m}	σ
Best	14.45	2.17	12.95	3.02	10.05	4.21	9.10	1.68	8.35	3.35	7.20	4.13
Worst	25.70	2.13	21.65	3.57	14.25	4.88	14.65	1.52	12.45	3.83	10.90	4.48

Table 4 shows the frequency of core migrations performed by both the optimizations over the entire lifetime of the simulation i.e. 2000 seconds. It can be clearly seen that the 2-otr optimization migrates very frequently as compared to the count-qos-packets optimization. With a 2-otr based core migration optimization, the core only waits for latest delay samples (i.e. MAX-QoS-SAMPLES or 1 ack packet per OTR) from its 2 on tree routers. However, for a count-qos-packets optimization, the core has to either wait for a minimum of MAX-ACK-PACKETS or one ack-packet for every on-tree-router it may have currently as its neighbors. It is to be noted here that these observations were obtained with $\alpha = 0.8$, MAX-ACK-PACKETS = 3, and MAX-QoS-SAMPLES = 5. A similar effect of infrequent core migrations has been observed for reduced values of α such as 0.3 and 0.5. Thus count-qos-packets based optimization is not as responsive to network dynamics as compared to 2-otr optimization. From tables 2 and 3, it is therefore observed that 2-otr optimization performs well in sparse deployment areas such as 3000m×800m, whereas count-qos-packets optimization performs well in relatively dense areas such as 1500m×800m.

As mentioned earlier, the improvements in delay performance are not appreciable and are very modest. There are a number of reasons for this and these can be attributed to the design philosophy of our protocol. The technique of moving the core closer to the leaves of the tree by way of core migration is only a best effort scheme. It is not enough for QoS provisioning since we are not considering the path from the source to the new core which varies with every core migration. We only do 1-hop core migration without reconfiguring specific tree-branches that can expect a higher delay. Our scheme uses a fixed value of $delay_{threshold}$ and $delay_{target}$ to take care of incremental improvements in

delay for any 1-hop migration and network mobility rates. Currently, the core makes a migration decision over a history of samples using some fixed value of α and migrates to a branch with the largest delay. With high relative mobility, α can be set to a low value, say 0.2, and otherwise it can set a higher value, say 0.8. Also, we haven't incorporated any scheme that identifies the benefits of migration before moving the core to its next hop neighbor. We expect that an integration of proper prediction and adaptation schemes to our best-effort protocol would provide much improved performance.

5 Conclusions and Future Work

In this paper we have proposed a simple and efficient core migration protocol for providing QoS in MANET. The objective of core migration is to bring the core closer to the leaves of the tree that experience relatively higher delays. By dynamically migrating the core hop-by-hop, the protocol tries to meet the desired QoS specified by the multicast application. In the worst case where the QoS cannot be met, the protocol aims to balance the delays across all the subtree branches of the core. Extensive simulations conducted over three deployment areas suggest that core migration provides an additional benefit in terms of increasing the percentage packets meeting various QoS delay requirements.

As mentioned earlier, we are looking forward to understanding the dynamic relationship between node mobility, application specified qos target, qos threshold, and α . We believe that such an understanding would provide novel insights into the design and integration of various adaptation and prediction schemes that our protocol needs. The best way of doing this may be to develop our own multicast routing protocol that inherently supports the core migration protocol via application layer input (API). We would also like to use the concept of mobility prediction to make the protocol intelligent.

Our approach of QoS-aware core migration on a single core based multicast tree can be extended to maintaining and migrating multiple cores in a very large network with a number of multicast sessions. Cluster based solutions are similar to multiple cores where clusterheads act as cores for their cluster members that also communicate in tree-like multicast fashion. However, solutions for these kinds of analogous architectures are challenging. Providing QoS on this architecture by borrowing some ideas from the core migration protocol seems a reasonable way our extending the work.

6 Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant ANI-0086020 and a summer dissertation fellowship from Wayne State University. We are thankful to Y. Zhu and his research group at the University of Carleton for making their MAODV code available online for reuse.

References

- [1] G. Ahn, A. Campbell, A. Veres, and L. Sun. Swan: Service differentiation in stateless wireless ad hoc networks. In *Proceedings of IEEE INFOCOM '2002*, 2002.
- [2] A. Banerjea, M. Faloutsos, and R. Pankaj. Designing QoSMIC: A quality of service sensitive multicast Internet protocol, 1998.
- [3] S. Blake, D. Black, M. Carson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services, Dec 1998.
- [4] Bluetooth Charter. Bluetooth Homepage. <http://www.bluetooth.com>.
- [5] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: An Overview, June 1994.
- [6] K. Carlberg and J. Crowcroft. Building shared trees using a one-to-many joining mechanism. *ACM Computer Communication Review*, pages 5–11, Jan 1997.
- [7] K. Carlberg and J. Crowcroft. Yet Another Multicast Routing Protocol: specification version 1, 1997.
- [8] S. Chakrabarti and A. Mishra. QoS Issues in Ad Hoc Networks. *IEEE Communications Magazine*, pages 142–148, Feb 2001.
- [9] K. Chen, S. H. Shah, and K. Nahrstedt. Cross Layer Design for Data Accessibility in Mobile Ad Hoc Networks. *Journal of Wireless Communications*, pages 49–75, 2002.
- [10] S. Chen and K. Nahrstedt. Distributed quality-of-service routing in ad-hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8), August 1999.
- [11] IEEE LAN MAN Standards Committee. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standard 802.11-1997, IEEE Computer Society, NY, USA, 1997.
- [12] M. Conti, G. Maselli, G. Turi, and S. Giordano. Cross-layering in mobile ad hoc network design. *IEEE Computer, special issue on Ad Hoc Networks*, 37(2):48–51, Feb 2004.

- [13] M. S. Corson. Issues in supporting quality of service in mobile ad hoc networks. In *Proceedings of the 5th Int. Workshop on Quality of Service (IWQOS '97)*, May 1997.
- [14] M. S. Corson and A. T. Campbell. Towards supporting quality of service in mobile ad hoc networks. In *Proceedings of the First Conference on Open Architecture and Network Programming, San Francisco*, Apr 1998.
- [15] C. Diot, W. Dabbous, and J. Crowcroft. Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms. *IEEE Journal on Selected Areas in Communications*, 15(3):277–290, April 1997.
- [16] M. Ilyas (editor), X. Chen, and J. Wu. *Handbook of Ad Hoc Wireless Networks*, chapter Multicasting techniques in mobile ad hoc networks. CRC Press, 2002.
- [17] E. Fleury, Y. Huang, and P. K. McKinley. On the Performance and Feasibility of Multicast Core Selection Heuristics. In *7th International Conference on Computer Communications and Networks, Lafayette, Louisiana*, October 1998.
- [18] S. Giordano and W. Lu. Challenges in mobile ad hoc networking. *IEEE Communications Magazine*, 39(6):129–129, Jun 2001.
- [19] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 64–71, Seattle, WA, 1999.
- [20] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.
- [21] M. Kochhal, L. Schwiebert, S. K. S. Gupta, and C. Jiao. An Efficient Core Migration Protocol for QoS in Mobile Ad hoc Networks. In *21st IEEE International Performance Computing and Communications (IPCCC '02), Phoenix*, pages 387–391, April 2002.
- [22] S. B. Lee and A. T. Campbell. Insignia: In-band signaling support for qos in mobile ad hoc networks. In *Proceedings of the 5th Int. Workshop on Mobile Multimedia Communications (MoMuc'98), Berlin*, Oct 1998.
- [23] S. B. Lee, A. Gahng-Seop, X. Zhang, and A. T. Campbell. INSIGNIA: An IP-Based Quality of Service Framework for Mobile Ad Hoc Networks. *Journal of Parallel and Distributed Computing (Academic Press)*, Special issue on *Wireless and Mobile Computing and Communications*, 60(4):374–406, April 2000.
- [24] C. R. Lin and J. S. Liu. QoS routing in ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, 17:1426–1438, 1999.
- [25] Mobile Ad hoc Networks (MANET) Charter. MANET Homepage. <http://www.ietf.org/html.charters/manet-charter.html>.

- [26] P. Mohapatra, C. Gui, and J. Li. Group Communications in Mobile Ad Hoc Networks. *IEEE Computer Magazine*, pages 52–59, Feb 2004.
- [27] P. Mohapatra, J. Li, and C. Gui. QoS in Mobile Ad hoc Networks. *IEEE Wireless Communications Magazine*, pages 44–52, June 2003.
- [28] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th annual ACM/IEEE Int. conference on Mobile computing and networking (MOBICOM '99)*, pages 151–162, 1999.
- [29] V. Park and J. Macker. Anycast routing for mobile networking. In *Proceedings of the Military Communication Conference (MILCOM '99)*, October 1999.
- [30] C. E. Perkins. *Ad Hoc Networking*. Addison Wesley, 2001.
- [31] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of IEEE WMCSA '99*, Feb. 1999.
- [32] G. J. Pottie and W. Kaiser. Wireless Sensor Networks. *Communications of the ACM*, 43(5):51–58, May 2000.
- [33] R. Ramanathan and M. Steenstrup. Hierarchically-organized, multihop mobile wireless networks for quality-of-service support. *Journal of Mobile Networks and Applications*, 3(1):101–119, 1998.
- [34] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 2002.
- [35] E. M. Royer and C. E. Perkins. Multicast operation of the Ad hoc On-demand Distance Vector Routing. In *Proceedings of the 5th ACM/IEEE Annual Conference on Mobile Computing and Networking*, pages 207–218, August 1999.
- [36] P. Sinha, R. Sivakumar, and V. Bharghavan. Cedar: A core-extraction distributed ad hoc routing algorithm. In *Proceedings of IEEE INFOCOM '99, New York*, pages 202–209, May 1999.
- [37] D. G. Thaler and C. V. Ravishankar. Distributed Center-Location Algorithms. *IEEE Journal on Selected Areas in Communications*, 15(3):291–303, April 1999.
- [38] B. Tuch. Development of WaveLAN, an ISM Band Wireless LAN. Technical Journal, Lucent Labs, July 1993.
- [39] USC/ISI, UCB, LBL, and Xerox PARC. ns2: network simulator. <http://www.isi.edu/nsnam/ns/>.
- [40] A. Veres, A. T. Campbell, M. Barry, and L-H. Sun. Supporting Service Differentiation in Wireless Packet Networks Using Distributed Control. *IEEE Journal on Selected Areas of Communication (JSAC)*, 19(10):2081–2093, October 2001.
- [41] B. Wang and C.-J. Hou. A Survey on Multicast Routing and its QoS Extension: Problems, Algorithms, and Protocols. *IEEE Network Magazine*, pages 22–36, Feb 2000.

- [42] Z. Wang and J. Crowcroft. QoS Routing for supporting resource reservation. *IEEE Journal on Selected areas in Communications*, Sep 1996.
- [43] Wi-Fi. Wi-Fi Homepage. <http://www.wi-fi.org>.
- [44] K. Wu and J. Harms. QoS Support in Mobile Ad hoc Networks. *Crossing Boundaries: an interdisciplinary journal, University of Alberta*, 1(1), Sept. 2001.
- [45] K. Xu, K. Tang, R. Bagrodia, M. Gerla, and M. Bereschinsky. Adaptive bandwidth management and qos provisioning in large scale ad hoc networks. In *Proceedings of the Military Communication Conference (MILCOM '03)*, 2003.
- [46] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network Magazine*, pages 8–18, Sep 1993.
- [47] Y. Zhu. Proactive Connection Maintenance in AODV and MAODV. Master's thesis, Carleton University, Ottawa, Ontario, Canada, August 2002.